

Contents

1 Routine/Function Prologues	2
1.0.1 maketiles_nongds.F90 (Source File: maketiles_nongds.F90)	2

1 Routine/Function Prologues

1.0.1 maketiles_nongds.F90 (Source File: maketiles_nongds.F90)

This primary goal of this routine is to determine tile space for MPI-based I/O

REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Major F90 and major structure revision
3 Jan 2000: Minor T=0 bug fix, should have no effect on output
8 Mar 2000: Brian Cosgrove; Initialized FGRD to 0 For Dec Alpha Runs
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
04 Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space
17 Oct 2003: Sujay Kumar ; Initial version of subsetting code

```

INTERFACE:

```
subroutine maketiles_nongds()
```

USES:

```

use lisdrv_module, only: lis, grid, glbgindex, tile
use grid_module
use spmdMod

```

CONTENTS:

```

if ( masterproc ) then
    if(lis%d%kgds(42) > lis%d%lnc .or. &
       lis%d%kgds(43) > lis%d%lnr)  then !using a subdomain
        gnc = lis%d%kgds(42)
        gnr = lis%d%kgds(43)
    else
        gnc = lis%d%lnc
        gnr = lis%d%lnr
    endif
    lis%d%gnc = gnc
    lis%d%gnr = gnr

    allocate(lat(gnc,gnr), stat=ierr)
    call check_error(ierr,'Error allocating lat.',iam)

    allocate(lon(gnc,gnr), stat=ierr)
    call check_error(ierr,'Error allocating lon.',iam)

    allocate(mask(gnc, gnr), stat=ierr)
    call check_error(ierr,'Error allocating mask.',iam)

    nchp = lis%d%glbnch

```

```

print*, 'MSG: maketiles -- Reading ',trim(lis%p%myfile), &
      ' (',iam,')'
open(30,file=lis%p%myfile,form='unformatted',status='old')
read(30) lat
read(30) lon
read(30) mask
close(30)
print*, 'MSG: maketiles -- Done reading ',trim(lis%p%myfile), &
      ' (',iam,')'
allocate(locallat(lis%d%lnc,lis%d%lnr))
allocate(locallon(lis%d%lnc,lis%d%lnr))
allocate(localmask(lis%d%lnc,lis%d%lnr))
localmask = 0

do r=1,gnr
  do c=1,gnc
    if(lat(c,r)*1000.ge.lis%d%kgds(4).and. &
       lat(c,r)*1000.le.lis%d%kgds(7).and. &
       lon(c,r)*1000.ge.lis%d%kgds(5).and. &
       lon(c,r)*1000.le.lis%d%kgds(8)) then
      rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
                /lis%d%kgds(9)
      cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
                /lis%d%kgds(10)
      locallat(cindex,rindex) = lat(c,r)
      locallon(cindex,rindex) = lon(c,r)
      localmask(cindex,rindex) = mask(c,r)
    endif
  end do
end do
deallocate(mask)
call absoft_release_cache()

if(lis%f%ecor .eq. 1) then
  allocate(tmpelev(gnc*gnr),stat=ierr)
  call check_error(ierr,'Error allocating elevation.',iam)

  allocate(elevdiff(lis%d%lnc,lis%d%lnr), stat=ierr)
  call check_error(ierr,'Error allocating elev diff.',iam)

  elevdiff = 0.0
  tmpelev = 0.0
  open(21,file=lis%p%elevfile,form='unformatted',&
        status='old',iostat=ierr)
  if (ierr /= 0) then
    write(79,*) "stop: problem opening elevation difference file"
    write(79,*) "try running without elevation correction option."
    print*, "stop: problem opening elevation difference file"

```

```

        print*, "try running without elevation correction option."
        call endrun
    else
        read (21) (tmpelev(ppp), ppp = 1,gnc*gnr)
    endif
    close(21)

    cc=0
    do r=1,lis%d%gnr
        do c=1,lis%d%gnc
            if(lat(c,r)*1000.ge.lis%d%kgds(4).and. &
               lat(c,r)*1000.le.lis%d%kgds(7).and. &
               lon(c,r)*1000.ge.lis%d%kgds(5).and. &
               lon(c,r)*1000.le.lis%d%kgds(8)) then
                rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
                           /lis%d%kgds(9)
                cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
                           /lis%d%kgds(10)
                elevdiff(cindex,rindex) = tmpelev(c+cc)
            endif
        enddo
        cc = cc + lis%d%gnc
    enddo
    print*, 'done reading elevation difference file..'
    deallocate(tmpelev)
    call absoft_release_cache()
endif

allocate(fgrd(lis%d%lnc,lis%d%lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating fgrd.',iam)

allocate(veg(gnc, gnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating veg.',iam)

fgrd = 0
!-----
! Select which tile-space veg. info to use (UMD or Koster)
!-----
print*, 'MSG: maketiles -- Reading ',trim(lis%p%vfile), &
      ' (',iam,')'
open(98,file=lis%p%vfile,form='unformatted')
read(98) lat
read(98) lon
do t = 1, lis%p%nt
    read(98) veg(:,:,t)
enddo
print*, 'MSG: maketiles -- Done reading ',trim(lis%p%vfile), &
      ' (',iam,')'

```

```

do r=1,gnr
  do c=1,gnc
    isum=0.0
    if(lat(c,r)*1000.ge.lis%d%kgds(4).and. &
       lat(c,r)*1000.le.lis%d%kgds(7).and. &
       lon(c,r)*1000.ge.lis%d%kgds(5).and. &
       lon(c,r)*1000.le.lis%d%kgds(8)) then
      rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
                 /lis%d%kgds(9)
      cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
                 /lis%d%kgds(10)
      do t=1,lis%p%nt
        isum=isum+veg(c,r,t) !recompute ISUM without water points
      enddo
      do t=1,lis%p%nt
        fgrd(cindex,rindex,t)=0.0
        if(isum.gt.0) fgrd(cindex,rindex,t)=veg(c,r,t)/isum
      enddo
    end if
  enddo
enddo
close(98)

print*, 'MSG: maketiles -- Done calculations with ',&
trim(lis%p%vfile), &
' (',iam,')
call absoft_release_cache()

allocate(tsum(lis%d%lnc, lis%d%lnr), stat=ierr)
call check_error(ierr,'Error allocating tsum.',iam)
tsum = 0.0
!-----
! Exclude tiles with MINA (minimum tile grid area),
! normalize remaining tiles to 100%
!-----
do r=1,lis%d%lnr
  do c=1,lis%d%lnc
    rsum=0.0
    do t=1,lis%p%nt
      if(fgrd(c,r,t).lt.lis%d%mina)then
        fgrd(c,r,t)=0.0
      endif
      rsum=rsum+fgrd(c,r,t)
    enddo
!-----
! renormalize veg fractions within a grid to 1
!-----
    if(rsum.gt.0.0) then

```

```

        do t=1,lis%p%nt
            if(rsum.gt.0.0)fgrd(c,r,t)=fgrd(c,r,t)/rsum
        enddo

        rsum=0.0
        do t=1,lis%p%nt
            rsum=rsum+fgrd(c,r,t)
        enddo

        if(rsum.lt.0.9999.or.rsum.gt.1.0001)then
            write(*,*) 'Error1 in vegetation tiles',rsum,c,r
        endif
        endif
    enddo
enddo

allocate(pveg(lis%d%lnc,lis%d%lnr,lis%p%nt), stat=ierr)
call check_error(ierr,'Error allocating pveg.',iam)

!-----
! Exclude tiles with MAXT (Maximum Tiles per grid),
!   normalize remaining tiles to 100%
! Determine the grid predominance order of the tiles
! PVEG(NT) will contain the predominance order of tiles
!-----

        do r=1,lis%d%lnr
            do c=1,lis%d%lnc
                do t=1,lis%p%nt
                    fvt(t)=fgrd(c,r,t)
                    pveg(c,r,t)=0
                enddo
                do i=1,lis%p%nt
                    max=0.0
                    t=0
                    do j=1,lis%p%nt
                        if(fvt(j).gt.max)then
                            if(fgrd(c,r,j).gt.0) then
                                max=fvt(j)
                                t=j
                            endif
                        endif
                    enddo
                    if(t.gt.0) then
                        pveg(c,r,t)=i
                        fvt(t)=-999.0
                    endif
                enddo
            enddo
        enddo
    enddo
enddo

```

```

!-----
! Impose MAXT Cutoff
!-----

      do r=1,lis%d%lnr
        do c=1,lis%d%lnc
          rsum=0.0
          do t=1,lis%p%nt
            if(pveg(c,r,t).lt.1) then
              fgrd(c,r,t)=0.0
              pveg(c,r,t)=0
            endif
            if(pveg(c,r,t).gt.lis%d%maxt) then
              fgrd(c,r,t)=0.0
              pveg(c,r,t)=0
            endif
            rsum=rsum+fgrd(c,r,t)
          enddo
        enddo
      !-----

      ! renormalize veg fractions within a grid to 1
      !-----

      if(rsum.gt.0.0) then
        do t=1,lis%p%nt
          if(rsum.gt.0.0)fgrd(c,r,t)= fgrd(c,r,t)/rsum
        enddo

        rsum=0.0
        do t=1,lis%p%nt
          rsum=rsum+ fgrd(c,r,t) !recalculate rsum to check
        enddo
        tsum(c,r)=rsum

        if(rsum.lt.0.9999.or.rsum.gt.1.0001)then !check renormalization
          write(*,*) 'Error2 in vegetation tiles',rsum,c,r
        endif
      endif
      enddo
    enddo
    deallocate(pveg)
    call absoft_release_cache()

    landnveg = 5
  !-----

  ! Make Tile Space
  !-----
```

```

lis%d%glbnch=0
do t=1,lis%p%nt
  do r=1,lis%d%lnr
    do c=1,lis%d%lnc
```

```

        if(localmask(c,r).gt.0.99.and. &
           localmask(c,r).lt.3.01)then !we have land
           if(fgrd(c,r,t).gt.0.0)then
               lis%d%glbnch=lis%d%glbnch+1
           endif
           if(tsum(c,r).eq.0.0.and.t.eq.landnveg)then
               lis%d%glbnch=lis%d%glbnch+1
           endif
       endif
   enddo
enddo
enddo

print*, 'DBG: maketiles -- glbnch',lis%d%glbnch,' (',iam,',')

allocate(tile(lis%d%glbnch))

lis%d%glbngrid=0
do r=1,lis%d%lnr
    do c=1,lis%d%lnc
        if(localmask(c,r).gt.0.99 .and. &
           localmask(c,r).lt.3.01) then
            lis%d%glbngrid=lis%d%glbngrid+1
        endif
    enddo
enddo
count = 1
print*, 'DBG: maketiles1 -- glbnch',lis%d%glbnch,' (',iam,',')
allocate(grid(lis%d%glbngrid))
allocate(glbgindex(lis%d%lnc, lis%d%lnr))
print*, 'DBG: maketiles2 -- glbnch',lis%d%glbnch,' (',iam,',')
do r=1,lis%d%lnr
    do c=1,lis%d%lnc
        glbgindex(c,r) = -1
        if(localmask(c,r).gt.0.99 .and. &
           localmask(c,r).lt.3.01) then
            grid(count)%lat = locallat(c,r)
            grid(count)%lon = locallon(c,r)
            grid(count)%fgrd = fgrd(c,r,:)
            glbgindex(c,r) = count
            count = count+1
        endif
    enddo
enddo
deallocate(locallat,stat=ierr)
call absoft_release_cache()
call check_error(ierr,'Error allocating glblat.',iam)
deallocate(locallon, stat=ierr)
call check_error(ierr,'Error allocating glblon.',iam)

```

```

    print*, 'DBG: maketiles3 -- glbnch',lis%d%glbnch,' (',iam,')'
!-----
!     For writing dominant Vegetation types
!-----

    if(lis%o%wparam .eq.1) then
        allocate(domveg(lis%d%lnc,lis%d%lnr))
        domveg = 0
    endif
    count = 0
    do r=1,gnr
        do c=1,gnc
            if(lat(c,r)*1000.ge.lis%d%kgds(4).and. &
               lat(c,r)*1000.le.lis%d%kgds(7).and. &
               lon(c,r)*1000.ge.lis%d%kgds(5).and. &
               lon(c,r)*1000.le.lis%d%kgds(8)) then
                rindex = r - (lis%d%kgds(4)-lis%d%kgds(44)) &
                           /lis%d%kgds(9)
                cindex = c - (lis%d%kgds(5)-lis%d%kgds(45)) &
                           /lis%d%kgds(10)
                do t=1,lis%p%nt
                    if(localmask(cindex,rindex).gt.0.99.and. &
                       localmask(cindex,rindex).lt.3.01)then
                        if(fgrd(cindex,rindex,t).gt.0.0)then
                            count = count+1
                            tile(count)%row=r
                            tile(count)%col=c
                            tile(count)%index = glbgindex(cindex,rindex)
                            tile(count)%vegt=t
                            if(lis%o%wparam.eq.1) then
                                domveg(cindex,rindex) = t*1.0
                            endif
                            tile(count)%fgrd=fgrd(cindex,rindex,t)
                            if(lis%f%ecor.eq.1) &
                                tile(count)%elev=elevdiff(cindex,rindex)
                        endif
                    endif
                enddo
            endif
        enddo
    enddo
!-----
! What if we have land without vegetation assigned
!-----

        if(tsum(cindex,rindex).eq.0.0.and.t.eq.landnveg)then
            count=count+1
            tile(count)%row=r
            tile(count)%col=c
            tile(count)%index = glbgindex(cindex,rindex)
            tile(count)%vegt=t
            if(lis%o%wparam.eq.1) then
                domveg(cindex,rindex) = t*1.0
            endif
            tile(count)%fgrd=1.0

```

```

        if(lis%f%ecor.eq.1) &
            tile(count)%elev=elevdiff(cindex,rindex)
        endif
    endif
enddo
endif
enddo
endif
if(lis%o%wparam.eq.1) then
    open(32,file="domvegtype.bin",form='unformatted')
    write(32) domveg
    close(32)
    deallocate(domveg)
endif
if(lis%f%ecor.eq.1)  deallocate(elevdiff)
deallocate(lat)
deallocate(lon)
print*, 'DBG: maketiles4 -- glbnch',lis%d%glbnch,' (',iam,)'
deallocate(localmask, stat=ierr)
call check_error(ierr,'Error allocating glbmask',iam)
deallocate(fgrd, stat=ierr)
call check_error(ierr,'Error allocating glbfgrd',iam)
deallocate(tsum, stat=ierr)
call check_error(ierr,'Error allocating glbtsum.',iam)
call absoft_release_cache()

WRITE(*,*) 'MSG: maketiles -- Size of Tile Dimension:',NCHP, &
    ' (',iam,)'
WRITE(*,*) 'MSG: maketiles -- Actual Number of Tiles:', &
    LIS%D%GLBNCH,' (',iam,)'
WRITE(*,*)

WRITE(*,*) 'MSG: maketiles -- Size of Grid Dimension:', &
    lis%d%glbngrid,' (',iam,)'
WRITE(*,*)

WRITE(79,*)'MSG: maketiles -- Size of Tile Dimension:',NCHP, &
    ' (',iam,)'
WRITE(79,*)'MSG: maketiles -- Actual Number of Tiles:', &
    LIS%D%GLBNCH,' (',iam,)'
WRITE(79,*)

endif
print*, 'MSG: maketiles -- done', ' (',iam,)'
return

```